Introduction

With the introduction of Acron RPN Calculator v3.0, the UI is now fully customizable. Layouts can be authored as xml and saved with the .acronrpn file extension. When the files are opened in Acron RPN Calculator, the new customized layout will be applied.

Acron RPN expects a layout .acronrpn file to have the following basic structure. Each section of the document will be discussed in detail below.

```
<?xml version="1.0" encoding="utf-8"?>
<root>
      <appearance>
            ...
      </appearance>
      <layout ...>
            ...
      </layout>
      <behavior>
            ...
      </behavior>
      <values>
            ...
      </values>
</root>
```

Appearance

The <appearance> section defines all the drawing primitives that will be used in the layout. This includes colors, fonts, brushes, pens, scales, and styles.

Everything in the <appearance> section is processed sequentially. IDs must be defined before there is a ref to them.

Colors

Key Name

color

Attributes

ID	Unique identifier used by other objects to reference this color
value	Hexadecimal value of the color. Formatted as 0xAARRGGBB.
ref	Reference to the ID of another color. Makes this color the same as the ref color.

Note: Either value or ref must be defined. If both are defined, value will be used.

Fonts

Key Name

font

Attributes

ID	Unique identifier used by other objects to reference this font
family	Name of the font family. Roboto fonts are guaranteed to be available; other fonts must be installed on the system.
style	Font style. Valid values are 'regular', 'bold', 'italic', and 'bold italic'.
size	Size of font in points. Must be a positive floating-point number.
ref	Reference to the ID of another font. If family, style, or size are not defined, those values from the ref font will be used.

Note: Either family, style and size must be defined, or ref must be defined.

Brushes

Key Name

brush

Attributes

ID	Unique identifier used by other objects to reference this brush
ref	Reference to the ID of another brush.

Subkeys

color	Color of the brush. See Colors section above.

Note: Either color subkey or ref must be defined. If both are defined, color subkey will be used.

Pens

Key Name

pen

Attributes

ID	Unique identifier used by other objects to reference this pen
size	Size in points of the pen.
dashStyle	Dash style of the pen. Valid values are 'solid', 'dash', 'dot', 'dashDot', and 'dashDotDot'. If no value is provided, 'solid' will be used.
ref	Reference to the ID of another pen. If size, dashStyle, or color are not defined, those values from the ref pen will be used.

Subkeys

color	Color of the pen. See Colors section above.

Note: Either size and color subkey must be defined, or ref must be defined

Scales

Scales provide a way of ensuring that various text elements (usually button labels) can both zoom as large as possible to fill their container, and all be the same font size. Acron RPN will find the correct zoom to make the largest text element fit its available space, then scale all other elements that use the same scale object to match.

Key Name

scale

Attributes

ID	Unique identifier used by other objects to reference this scale
max	Reference to another scale to use as a maximum scale. This scale will be allowed to be smaller than max, but will not be allowed to be larger than max. Ignored if ref is defined.
ref	Reference to the ID of another scale. Makes this scale the same as the reference scale.

Styles

Styles provide the collection of fonts, brushes, pens colors, and scales necessary to draw UI elements.

Key Name

style

Attributes

ID	Unique identifier used by other objects to reference this style
padding	Space in points between a text element and the border of the container. Must be a positive floating-point value.
ref	Reference to the ID of another style. Any undefined fonts, pens, etc. in this style will be taken from the ref style.

Subkeys

normalFont	Font used for everything except based-integer numbers. See Fonts section above.
integerFont	Font used for based-integer numbers. If omitted, normalFont will be used. See Fonts section above.
foreBrush	Brush used for drawing filled foreground items. If omitted, will create a brush using the forePen color. See Brushes section above.
forePen	Pen used for drawing foreground items. See Pens section above.
backBrush	Brush used to draw the background of the current object. See Brushes section above.
borderBrush	Brush used to draw filled border shapes. If omitted, will create a brush using the borderPen color. See Brushes section above.
borderPen	Pen used for drawing the border around the current object. See Pens section above.
highlightBrush	Brush used to draw the background when the current object is highlighted. If omitted, the backBrush will be used. See Brushes section above.
scale	Scale used to synchronize the text size of all objects using this style. See Scales section above.

Note: Either normalFont, forePen, backBrush, borderPen, scale, and padding must be defined, or ref must be defined.

Regions

Multiple appearance keys can be surrounded by a `<region>` block to group them together. These regions can then be collapsed in and xml editor, making it easier to navigate your document. They have no impact on the functionality of your layout.

Key Name

region

Layout

The <layout> section defines the size and location of UI elements.

Attributes

orientation	Orientation the application will be displayed on your device. Valid values are 'portrait' and 'landscape'. Default value is 'portrait'.
numberFormat	Default number format for displaying answers. Value is a bitmask combination of:

PrecisionMask	0xF	The low order byte gives the number of decimal places
Fixed	0x100	True if trailing zeros should be displayed after the decimal
Scientific	0x200	True if decimals should be shown in scientific notation (Cannot coexist with Engineering)
Engineering	0x400	True if decimals should be shown in engineering notation (Cannot coexist with Scientific)
Perfect	0x800	True if perfect answer should be displayed (if available). False will cause a decimal answer to always be displayed.

base	Default base that should be used when in based-integer mode. Valid values are '2', '10', and '16'.
bits	Default bit size that should be used when in based-integer mode. Valid values are '1', '8', '16', '32', and '64'.
scientific	Whether scientific or based-integer mode should be set. Valid values are 'true' (start in scientific mode) and 'false' (start in based-integer mode).

Labels

Labels display text or math equations on the screen.

Key Name

label

Attributes

ID	Unique identifier used by other objects to reference this label
label	Text to display on the label
superscript	Text to display in superscript. If label is defined, the superscript text is displayed to the right of the label text.
subscript	Text to display in subscript. If label is defined, the subscript text is displayed to the right of the label text.
parameter	Additional text to display. Text is displayed in the same font size and vertical position as label, but is displayed to the right of label, superscript, and subscript (if defined).

dynamic	Text will be generated based on the calculator state. Valid values are 'ConsoleText', 'EquationX', 'EquationY', 'EquationZ', 'EquationT', 'ResultX', 'ResultY', 'ResultZ', and 'ResultT'. If dynamic is defined, label, superscript, subscript, and parameter are ignored.
horizontalAlignment	How the label is positioned horizontally in its container. Valid values are 'left', 'center', 'right', 'fit', 'leftFit', and 'rightFit'. Default value is 'fit'.
verticalAlignment	How the label is positioned vertically in its container. Valid values are 'top', 'center', 'bottom', 'fit', 'topFit', and 'bottomFit'. Default value is 'fit'.
style	Style to use when drawing the label. Can be inherited from parent keys, but must be defined on some ancestor of label.

Subkeys

math	Labels can contain one math item to display. This will be positioned exactly as the label attribute would be. If a math subkey is defined, the label attribute will not be used.

Buttons

Key Name

button

Attributes

ID	Unique identifier used by other objects to reference this button
label	Text to display on the button
superscript	Text to display in superscript. If label is defined, the superscript text is displayed to the right of the label text.
subscript	Text to display in subscript. If label is defined, the subscript text is displayed to the right of the label text.
parameter	Additional text to display. Text is displayed in the same font size and vertical position as label, but is displayed to the right of label, superscript, and subscript (if defined).
enabled	Whether the button is enabled (i.e. can be clicked). Valid values are 'true' and 'false'. Default is 'true'.
enableLongPressWhileDisabled	Whether the button can be long-pressed while it is disabled. Valid values are 'true' and 'false'. Default is 'false'.
pageFoldover	Whether the upper-right corner of the button is shaded as if folded. Layouts created by Acron use this effect to indicate that the button can be long-pressed, however this is not a requirement. Valid values are 'true' and 'false'. Default value is 'false'.

style	Style to use when drawing the button. Can be inherited from parent keys, but must be defined on some ancestor of button.
action	Action to perform when the button is clicked. Valid values are any StackAction or MacroAction from Appendix 1, or any function or subroutine defined in the values section. This is a convenient shortcut to bind the most frequently used actions to buttons without going through the complexity of the behavior section.
longPressAction	Action to perform when the button is long-pressed. Valid values are any StackAction or MacroAction from Appendix 1, or any function or subroutine defined in the values section.
activatePane	ID of a pane on a panel to display when the button is clicked. Must be paired with onPanel.
onPanel	ID of a panel for which the pane is being activated. Must be paired with activatePane.
expandPane	ID of a pane on a foldaway to expand when the button is clicked. Must be paired with onFoldaway and cannot coexist with collapsePane, togglePane, soloPane, or toggleSoloPane.
collapsePane	ID of a pane on a foldaway to collapse when the button is clicked. Must be paired with onFoldaway and cannot coexist with expandPane, togglePane, soloPane, or toggleSoloPane.
togglePane	ID of a pane on a foldaway to toggle when the button is clicked. Must be paired with onFoldaway and cannot coexist with expandPane, collapsePane, soloPane, or toggleSoloPane.
soloPane	ID of a pane on a foldaway to expand when the button is clicked; all other expandable panes will be collapsed. Must be paired with onFoldaway and cannot coexist with expandPane, collapsePane, togglePane, or toggleSoloPane.
toggleSoloPane	ID of a pane on a foldaway to toggle when the button is clicked; all other expandable panes will be collapsed. Must be paired with onFoldaway and cannot coexist with expandPane, collapsePane, togglePane, or soloPane.
onFoldaway	ID of a foldaway for which the pane is being manipulated. Must be paired with expandPane, collapsePane, togglePane, soloPane, or toggleSoloPane.
constant_push	ID of a constant to push to the stack. If there is a value on the console, the constant will be multiplied by it.
constant_multiply	ID of a constant to multiply by either the value on the console, or the top value on the stack. If there is no value on the stack, no action will be performed. Useful for creating unit conversions.

constant_inverse	ID of a constant to divide either the value on the console, or the top value on the stack. If there is no value on the stack, no action will be performed. Useful for creating the inverse unit conversion of constant_multiply without defining a new constant.
variable_push	ID of a variable to push to the stack. If there is a value on the console, the variable will be multiplied by it.
variable_store	ID of a variable to store either the value on the console, or the top value on the stack. If there is no value on the stack, no action will be performed.

Subkeys

math	Buttons can contain one math item to display. This will be positioned exactly as the label attribute would be. If a math subkey is defined, the label attribute will not be used. Cannot coexist with a UI subkey.
UI item	Buttons can contain one UI item (label, array, etc.) to display. This will fill the entire button space up to the padding. If a UI subkey is defined, label, superscript, subscript, and parameter will be ignored. Cannot coexist with a math subkey.

Arrays

Arrays are horizontal or vertical groups of UI elements. You can create grids of elements by creating vertical arrays of horizontal arrays (or horizontal arrays of vertical arrays).

Key Name

array

Attributes

ID	Unique identifier used by other objects to reference this array
alignment	**[required]** Whether this is a horizontal or vertical array. Valid values are ‘horizontal’ and ‘vertical’.

Subkey Attributes

size	Amount of the overall width (horizontal) or height (vertical) that this item should consume. Sum of the sizes of all subkeys must be 1.0. Space will be evenly divided for all subkeys that omit the size attribute.

Panels

Panels fill their entire space with one of their subkey panes. The active pane can be switched by a activatePane/onPanel attribute pair on a button, or from an event in the behavior section.

Key Name

panel

Attributes

ID	Unique identifier used by other objects to reference this panel

Subkey Attributes

ID	Only subkeys with an ID can be activated in the panel

Foldaways

Foldaways have a dynamically sized primary subkey pane, and fixed sized secondary subkey panes that can be turned on and off.

Key Name

foldaway

Attributes

ID	Unique identifier used by other objects to reference this array
alignment	**[required]** What side of the foldaway the secondary panes will be expanded from. Valid values are ‘left’, ‘right’, ‘top’, and ‘bottom’.
stacking	Whether only one secondary subkey should be displayed at a time. If true, panes will be stacked in the order they are defined, starting at the edge declared in ‘alignment’. If false, only the highest priority visible pane will be displayed. Valid values are ‘true’ and ‘false’. Default is ‘false’.

Subkey Attributes

ID	Only secondary subkeys with an ID can be activated in the foldaway
size	**[required]** Amount of the overall width or height of the foldaway that this subkey should consume. Sum of the sizes of visible subkeys should not exceed 1.0. Cannot exist on primary subkey; mandatory on secondary subkeys.
expanded	Whether the pane should be expanded by default. Valid values are ‘true’ and ‘false’. Default is ‘false’.

Stack

Displays the stack. There should be no more than one stack element in a layout.

Key Name

stack

Attributes

equationStyle	Style to use when drawing the equation side of the stack. Can be inherited from parent keys, but must be defined on some ancestor of stack.
resultsStyle	Style to use when drawing the results side of the stack. Can be inherited from parent keys, but must be defined on some ancestor of stack.
consoleStyle	Style to use when drawing the console of the stack. Can be inherited from parent keys, but must be defined on some ancestor of stack.
highlightBrush	Brush used to highlight the background of selected items on the stack. If omitted, the highlightBrush of equationStyle will be used.
highlightPen	Pen used to draw a border around the selected item on the stack. If omitted, the highlightPen of the consoleStyle will be used.
dividerPen	Pen used to draw the divider line between items on the stack. If omitted, the forePen of the consoleStyle will be used.
scrollbarBrush	Brush used to draw the stack scrollbar (visible only while scrolling). If omitted, the foreBrush of the consoleStyle will be used.
thousandSuffix	Word label displayed in alternate view. Can be any text value, but recommended values are " thousand" or "k". Default value is " thousand".
millionSuffix	Word label displayed in alternate view. Can be any text value, but recommended values are " million" or "M". Default value is " million".
billionSuffix	Word label displayed in alternate view. Can be any text value, but recommended values are " billion", " milliard", or "G". Default value is " billion".
trillionSuffix	Word label displayed in alternate view. Can be any text value, but recommended values are " trillion", " billion", or "T". Default value is " trillion".
quadrillionSuffix	Word label displayed in alternate view. Can be any text value, but recommended values are " quadrillion", " billiard", or "P". Default value is " quadrillion".
quintillionSuffix	Word label displayed in alternate view. Can be any text value, but recommended values are " quintillion", " trillion", or "E". Default value is " quintillion".

Macro

Displays the macro. There should be no more than one macro element in a layout.

Key Name

macro

Attributes

equationStyle	Style to use when drawing the equation side of the macro. Can be inherited from parent keys, but must be defined on some ancestor of macro.

resultsStyle	Style to use when drawing the results side of the macro. Can be inherited from parent keys, but must be defined on some ancestor of macro.
consoleStyle	Style to use when drawing the console of the macro. Can be inherited from parent keys, but must be defined on some ancestor of macro.
activeHighlightBrush	Brush used to highlight the background of the active cell in the macro. If omitted, the backBrush of the consoleStyle will be used.
matchingHighlightBrush	Brush used to highlight the background of matching cells in the macro. If omitted, the backBrush of the consoleStyle will be used.
enabledHighlightBrush	Brush used to highlight the background of the enabled cells in the macro. If omitted, the backBrush of the consoleStyle will be used.
disabledHighlightBrush	Brush used to highlight the background of disabled cells in the macro. If omitted, the backBrush of the consoleStyle will be used.
activeHighlightPen	Pen used to draw the border around the active cell in the macro. If omitted, the forePen of the consoleStyle will be used.
dividerPen	Pen used to draw the divider line between the console and macro area. If omitted, the forePen of the consoleStyle will be used.

Math

Creates an equation to use as the text for a button or label. Can be created using any of the Math objects from Appendix 1. The key name is the same as the object name. Must have the correct number of subkeys to create the object.

Behavior

The <behavior> section defines events and responses to those events.

Stack Events

ConsoleNoDecimal	A decimal point was just removed from the console
ConsoleHasDecimal	A decimal point was just added to the console and an exponential symbol is not on the console
ConsoleHasEE	An exponential symbol was just added to the console
ConsoleHasText	The console contains some text
ConsoleNoText	The console contains no text
StackSelected	An item on the stack is selected
StackNoSelect	No item on the stack is selected
StackChanged	A stack push, pop, or clear has occurred
BaseScientific	Stack is switched to scientific mode
BaseInteger	Stack is switched to based-integer mode

BaseBin	Stack is switched to binary based-integer mode
BaseDec	Stack is switched to decimal based-integer mode
BaseHex	Stack is switched to hexadecimal based-integer mode
BaseBool	Stack is switched to boolean based-integer mode
BaseByte	Stack is switched to byte based-integer mode
BaseWord	Stack is switched to word based-integer mode
BaseDword	Stack is switched to dword based-integer mode
BaseQword	Stack is switched to qword based-integer mode
BaseChanged	The stack base has changed. Includes scientific/integer, bin/dec/hex, and bool/byte/word/dword/qword.
DispFix	Stack display mode is switched to fixed decimal places
DispFloat	Stack display mode is switched to floating decimal places
DispSci	Stack display mode is switched to scientific notation
DispEng	Stack display mode is switched to engineering notation
DispAuto	Stack display mode is switched to automatic notation. This will automatically switch between normal and scientific notation as needed.
DispPerfect	Stack display mode is switched to display mathematically perfect answers (when possible)
DispApprox	Stack display mode is switched to always display decimal answers, even when a perfect answer is known.
DispSigFig[0-15]	Stack display mode has changed to the specified number of significant digits
DispChanged	Stack display mode has changed. Includes fix/float, sci/eng/auto, perfect/approx, and significant figures.
StatsAvailable	Statistics registers have been populated with statistics data
StatsUnavailable	Statistics registers have been cleared

Macro Events

EnterMacroMode	Application has entered macro mode
ExitMacroMode	Application has exited macro mode
MacroNoDecimal	A decimal point was just removed from the macro console
MacroHasDecimal	A decimal point was just added to the macro console and an exponential symbol is not on the macro console
MacroHasEE	An exponential symbol was just added to the macro console
MacroScientific	Selected cell in macro is in scientific mode
MacroInteger	Selected cell in macro is in based-integer mode

MacroBin	Selected cell in macro is in binary based-integer mode
MacroDec	Selected cell in macro is in decimal based-integer mode
MacroHex	Selected cell in macro is in hexadecimal based-integer mode
MacroBool	Selected cell in macro is in boolean based-integer mode
MacroByte	Selected cell in macro is in byte based-integer mode
MacroWord	Selected cell in macro is in word based-integer mode
MacroDword	Selected cell in macro is in dword based-integer mode
MacroQword	Selected cell in macro is in qword based-integer mode
SplitAvailable	The selected cell in macro can be split (i.e. it has matching cells)
SplitUnavailable	The selected cell in macro cannot be split
ZoomAvailable	The macro is sufficiently large such that it had to be scaled to fit the screen. Using the MacroBox.Zoom would result in the macro zooming.
ZoomUnavailable	The entire macro fits on the screen. MacroBox.Zoom would have no effect.

UI Events

All UI events have the UI type (button, panel, foldaway, etc.) as the key name, a 'ref' attribute referring to the applicable item's ID, and an 'event' attribute specifying which of that item's events to respond to. For example:

```
<button ref="SinButton" event="click">
```

button click	Button was pressed
button longPress	Button was long-pressed
button enabled	Button became enabled
button disabled	Button became disabled
button enableChanged	Button's enabled state changed (i.e. went from enabled to disabled, or went from disabled to enabled).
panel changed	The panel changed its active pane
panel nochange	The panel was instructed to activate a pane that was already active
panel activated	The specified pane has become active. Requires attribute 'pane' to be set to the ID of one of the panel's panes.
panel deactivated	The specified pane has become inactive. Requires attribute 'pane' to be set to the ID of one of the panel's panes.
panel changed	The active state of the specified pane has changed (i.e. went from active to inactive, or from inactive to active). Requires attribute 'pane' to be set to the ID of one of the panel's panes.
foldaway expanded	The specified pane has become expanded. Requires attribute 'pane' to be set to the ID of one of the foldaway's panes.

foldaway collapsed	The specified pane has become collapsed. Requires attribute 'pane' to be set to the ID of one of the foldaway's panes.
foldaway expanded	The specified pane has become expanded. Requires attribute 'pane' to be set to the ID of one of the foldaway's panes.
foldaway expandedChanged	The expanded state of the specified pane has changed (i.e. went from expanded to collapsed, or from collapsed to expanded). Requires attribute 'pane' to be set to the ID of one of the foldaway's panes.

Math Responses

Any StackAction or MacroAction from Appendix 1 can be added as event responses.

UI Responses

All UI responses have the UI type (button, panel, foldaway, etc.) as the key name and a 'ref' attribute referring to the applicable item's ID. For example:

```
<button ref="SinButton" enabled="toggle"/>
```

button enabled	Enables or disables the specified button. Requires attribute 'enabled' to be set to 'true', 'false', or 'toggle'.
panel	Activates the specified pane. Requires attribute 'pane' to be set to the ID of one of the panel's panes.
foldaway collapseAll	Collapses all the secondary panes of the foldaway. Requires attribute 'collapseAll' to be set to 'true'.
foldaway expanded	Expands or collapses the specified pane. Requires attribute 'pane' to be set to the ID of one of the foldaway's panes. Requires attribute 'expanded' to be set to 'true', 'false', 'toggle', 'solo', or 'toggleSolo'.
scale reset	Reset a scale object to its maximum size. Useful when buttons or labels are resized larger, making it necessary to recalculate the maximum font size that fits. Requires attribute 'reset' to be set to 'true'.

Value Responses

All value responses have the value action (constant_push, variable_store, etc.) as the key name and a 'ref' attribute referring to the applicable value's ID. For example:

```
<variable_push ref="myVariableX"/>
```

constant_push	Pushes the specified constant to the stack. If a value is on the console, it will be multiplied by the constant.
constant_multiply	Multiplies the constant by the top of the stack (or the console if it isn't empty). If no value is available to multiply, no action will be performed. Useful for unit conversions. Optional attributes 'label', 'superscript', 'subscript', and 'parameter' can be used to re-label constants created by this event response.
constant_inverse	Multiplies the inverse of the constant by the top of the stack (or the console if it isn't empty). If no value is available to multiply, no action will be performed.

	Useful for unit reverse unit conversions. Optional attributes 'label', 'superscript', 'subscript', and 'parameter' can be used to re-label constants created by this event response.
variable_push	Pushes the specified variable to the stack. If a value is on the console, it will be multiplied by the variable.
variable_store	Stores the selected item, top of the stack, or value on the console to the specified variable.
function	Invokes the specified user-defined function
subroutine	Invokes the specified user-defined subroutine

Regions

Multiple event keys can be surrounded by a `<region>` block to group them together. These regions can then be collapsed in and xml editor, making it easier to navigate your document. They have no impact on the functionality of your layout.

Key Name

region

Values

The <values> section declares constants and variables.

Constants

Declares values that cannot be altered

Key Name

constant

Attributes

ID	Unique identifier used by other objects to reference this constant
label	Text to display for this constant
superscript	Text to display in superscript. If label is defined, the superscript text is displayed to the right of the label text.
subscript	Text to display in subscript. If label is defined, the subscript text is displayed to the right of the label text.
parameter	Additional text to display. Text is displayed in the same font size and vertical position as label, but is displayed to the right of label, superscript, and subscript (if defined).

integer	Perfect integer value for the constant. Cannot coexist with 'approximate', 'Bool', 'Int8', 'Int16', 'Int32', 'Int64', or math subkeys.
approximate	Floating point value for the constant. Cannot coexist with 'integer', 'Bool', 'Int8', 'Int16', 'Int32', 'Int64', or math subkeys.
Bool	1-bit based integer value for the constant. Cannot coexist with 'integer', 'approximate', 'Int8', 'Int16', 'Int32', 'Int64', or math subkeys.
Int8	8-bit based integer value for the constant. Cannot coexist with 'integer', 'approximate', 'Bool', 'Int16', 'Int32', 'Int64', or math subkeys.
Int16	16-bit based integer value for the constant. Cannot coexist with 'integer', 'approximate', 'Bool', 'Int8', 'Int32', 'Int64', or math subkeys.
Int32	32-bit based integer value for the constant. Cannot coexist with 'integer', 'approximate', 'Bool', 'Int8', 'Int16', 'Int64', or math subkeys.
Int64	64-bit based integer value for the constant. Cannot coexist with 'integer', 'approximate', 'Bool', 'Int8', 'Int16', 'Int32', or math subkeys.

Subkeys

math	Constants can have one math subkey to define their value. This allows for the creation of mathematically perfect constants like $\sqrt{2}$.

Variables

Declares variables that can be stored to, recalled, used in functions, etc.

Key Name

variable

Attributes

ID	Unique identifier used by other objects to reference this variable
label	Text to display for this variable
superscript	Text to display in superscript. If label is defined, the superscript text is displayed to the right of the label text.
subscript	Text to display in subscript. If label is defined, the subscript text is displayed to the right of the label text.
parameter	Additional text to display. Text is displayed in the same font size and vertical position as label, but is displayed to the right of label, superscript, and subscript (if defined).
integer	Perfect integer initial value for the variable. Cannot coexist with 'approximate', 'Bool', 'Int8', 'Int16', 'Int32', 'Int64', or math subkeys.

approximate	Floating point initial value for the constant. Cannot coexist with 'integer', 'Bool', 'Int8', 'Int16', 'Int32', 'Int64', or math subkeys.
Bool	1-bit based integer value for the constant. Cannot coexist with 'integer', 'approximate', 'Bool', 'Int8', 'Int16', 'Int32', 'Int64', or math subkeys.
Int8	8-bit based initial integer value for the variable. Cannot coexist with 'integer', 'approximate', 'Bool', 'Int16', 'Int32', 'Int64', or math subkeys.
Int16	16-bit based integer value for the variable. Cannot coexist with 'integer', 'approximate', 'Bool', 'Int8', 'Int32', 'Int64', or math subkeys.
Int32	32-bit based integer value for the variable. Cannot coexist with 'integer', 'approximate', 'Bool', 'Int8', 'Int16', 'Int64', or math subkeys.
Int64	64-bit based integer value for the variable. Cannot coexist with 'integer', 'approximate', 'Bool', 'Int8', 'Int16', 'Int32', or math subkeys.

Subkeys

math	Variables can have one math subkey to define their initial value. This allows for the creation of mathematically perfect values like $\sqrt{2}$.

Functions

Function allow you to wrap multiple StackActions from Appendix 1, and present them as a single built-in function. If any of StackActions is unable to complete, or does not complete in a timely manner, the user's device will vibrate to indicate the error, and the stack will be restored to its state prior to beginning the function.

Key Name

function

Attributes

ID	Unique identifier used by other objects to reference this function
label	Text to display for this function
superscript	Text to display in superscript. The superscript text is displayed to the right of the label text.
subscript	Text to display in subscript. The subscript text is displayed to the right of the label text.
parameters	The number of parameters that will be taken from the stack and used in this function. The function will not be allowed to pop more items off the stack than those requested in the parameters attribute. If the needed number of parameters is not available on the stack, the function will be aborted. All parameters to the function will be drawn as a comma separated list within the function parentheses. If attribute is not defined, 0 is assumed.

supportsMacros	Whether the parameters to this function can be modified in macro mode. If false, the entire function with all of its parameters will be treated as an unmodifiable constant. Valid values are 'true' and 'false'. Default is 'false'.
supportsSolver	Whether any variables within the function can be updated during MathWhere, MathZero, MathCritical, MathDerivitave, MathIntegral, MathSummation, or MathProduct calculations. If false, the entire function with all of its parameters will be treated as an unmodifiable constant. Valid values are 'true' and 'false'. Default is 'false'.

Subkeys

StackActions	Any of the StackActions from Appendix 1 can be added as a subkey to a function. Those StackActions will be invoked sequentially.
ifX, ifY, ifZ, ifT	Check whether the contents of the specified register meet specific criteria, and only invoke the subkeys if they do. Valid attribute names are 'exists', 'isZero', 'isComplex', 'isApproximate', 'isBasedInteger', 'isPerfectInteger', 'isPositive', 'isNegative', 'wasSelected', 'isSelected', and 'isType'. For 'isType', value must be a valid type from Appendix 1, or the ID of a custom function. For all other attributes, valid values are 'true' and 'false'.
ifSelection	Check whether the contents of the selected stack item meet specific criteria, and only invoke the subkeys if they do. Valid attribute names are 'exists', 'isZero', 'isComplex', 'isApproximate', 'isBasedInteger', 'isPerfectInteger', 'isPositive', 'isNegative', 'wasSelected', 'isSelected', and 'isType'. For 'isType', value must be a valid type from Appendix 1, or the ID of a custom function. For all other attributes, valid values are 'true' and 'false'.
ifVariable	Check whether the variable specified in 'ref' meets specific criteria, and only invoke the subkeys if it does. Valid attribute names are 'exists', 'isZero', 'isComplex', 'isApproximate', 'isBasedInteger', 'isPerfectInteger', 'isPositive', and 'isNegative'. For all attributes, valid values are 'true' and 'false'.
elifX, elifY, elifZ, elifT	Check whether the contents of the specified register meet specific criteria, and only invoke the subkeys if they do. Valid attribute names are 'exists', 'isZero', 'isComplex', 'isApproximate', 'isBasedInteger', 'isPerfectInteger', 'isPositive', 'isNegative', 'wasSelected', 'isSelected', and 'isType'. For 'isType', value must be a valid type from Appendix 1, or the ID of a custom function. For all other attributes, valid values are 'true' and 'false'. Must be immediately proceeded by an if_ or elif_.
elifSelection	Check whether the contents of the selected stack item meet specific criteria, and only invoke the subkeys if they do. Valid attribute names are 'exists', 'isZero', 'isComplex', 'isApproximate', 'isBasedInteger', 'isPerfectInteger', 'isPositive', 'isNegative', 'wasSelected', 'isSelected', and 'isType'. For 'isType', value must be a valid type from Appendix 1, or the ID of a custom function. For all other attributes, valid values are 'true' and 'false'. Must be immediately proceeded by an if_ or elif_.
elifVariable	Check whether the variable specified in 'ref' meets specific criteria, and only invoke the subkeys if it does. Valid attribute names are 'exists', 'isZero',

	'isComplex', 'isApproximate', 'isBasedInteger', 'isPerfectInteger', 'isPositive', and 'isNegative'. For all attributes, valid values are 'true' and 'false'. Must be immediately proceeded by an if_ or elif_.
else	Invokes the subkeys if the proceeding if_ or elif_ failed. Must be immediately proceeded by an if_ or elif_.
whileX, whileY, whileZ, whileT	Continously check whether the contents of the specified register meet specific criteria, and invoke the subkeys while they do. Valid attribute names are 'exists', 'isZero', 'isComplex', 'isApproximate', 'isBasedInteger', 'isPerfectInteger', 'isPositive', 'isNegative', 'wasSelected', 'isSelected', and 'isType'. For 'isType', value must be a valid type from Appendix 1, or the ID of a custom function. For all other attributes, valid values are 'true' and 'false'.
whileSelection	Continously check whether the contents of the selected stack item meet specific criteria, and invoke the subkeys while they do. Valid attribute names are 'exists', 'isZero', 'isComplex', 'isApproximate', 'isBasedInteger', 'isPerfectInteger', 'isPositive', 'isNegative', 'wasSelected', 'isSelected', and 'isType'. For 'isType', value must be a valid type from Appendix 1, or the ID of a custom function. For all other attributes, valid values are 'true' and 'false'.
whileVariable	Continously check whether the variable specified in 'ref' meets specific criteria, and invoke the subkeys while it does. Valid attribute names are 'exists', 'isZero', 'isComplex', 'isApproximate', 'isBasedInteger', 'isPerfectInteger', 'isPositive', and 'isNegative'. For all attributes, valid values are 'true' and 'false'.

Subroutines

Subroutines perform multiple StackActions from Appendix 1, but present them as if those actions were invoke individually (i.e. they are not wrapped the way functions do). If any of StackActions is unable to complete, or does not complete in a timely manner, the user's device will vibrate to indicate the error, and the stack will be restored to its state prior to beginning the subroutine.

Key Name

subroutine

Attributes

ID	Unique identifier used by other objects to reference this subroutine
parameters	The number of parameters that will be taken from the stack and used in this subroutine. The subroutine will not be allowed to pop more items off the stack than those requested in the parameters attribute. If the needed number of parameters is not available on the stack, the subroutine will be aborted. If the parameters attribute is not defined, subroutine will be allowed to consume any number of items from the stack, but trying to consume more than are available will abort the subroutine.

Subkeys

Same as function

Regions

Multiple value keys can be surrounded by a `<region>` block to group them together. These regions can then be collapsed in and xml editor, making it easier to navigate your document. They have no impact on the functionality of your layout.

Key Name

region

Appendix 1

StackBox.Abort

If called from a function or subroutine, will abort the entire call stack of all running functions/subroutines, and return the stack to its state before invoking the function/subroutine.

Parameters

none

StackBox.ApplyConsole

If there is a value on the console, pushes it to the stack. If the console is empty, has no effect.

Parameters

none

StackBox.ApplyDispToSelected

Applys the current display mode to the selected item on the stack. If no item is selected, applies to the top of the stack.

Parameters

None

StackBox.Backspace

Deletes one character from the console

Parameters

None

StackBox.Clear

Clears the console, the selected item on the stack, or the top item on the stack

Parameters

None

StackBox.ClearConsole

Clears the text on the console

Parameters

None

StackBox.ClearStack

Clears the console and all items on the stack

Parameters

None

StackBox.Decimal

Adds a decimal to the console, if one does not already exist, and the number format allows it.

Parameters

None

StackBox.Digit0

Adds a zero to the console (if the number format allows it)

Parameters

None

StackBox.Digit1

Adds a one to the console (if the number format allows it)

Parameters

None

StackBox.Digit2

Adds a two to the console (if the number format allows it)

Parameters

None

StackBox.Digit3

Adds a three to the console (if the number format allows it)

Parameters

None

StackBox.Digit4

Adds a four to the console (if the number format allows it)

Parameters

None

StackBox.Digit5

Adds a five to the console (if the number format allows it)

Parameters

None

StackBox.Digit6

Adds a six to the console (if the number format allows it)

Parameters

None

StackBox.Digit7

Adds a seven to the console (if the number format allows it)

Parameters

None

StackBox.Digit8

Adds an eight to the console (if the number format allows it)

Parameters

None

StackBox.Digit9

Adds a nine to the console (if the number format allows it)

Parameters

None

StackBox.DigitA

Adds a hexadecimal ‘A’ to the console (if the number format allows it)

Parameters

None

StackBox.DigitB

Adds a hexadecimal ‘B’ to the console (if the number format allows it)

Parameters

None

StackBox.DigitC

Adds a hexadecimal ‘C’ to the console (if the number format allows it)

Parameters

None

StackBox.DigitD

Adds a hexadecimal ‘D’ to the console (if the number format allows it)

Parameters

None

StackBox.DigitE

Adds a hexadecimal ‘E’ to the console (if the number format allows it)

Parameters

None

StackBox.DigitF

Adds a hexadecimal ‘F’ to the console (if the number format allows it)

Parameters

None

StackBox.DownArrow

Moves the stack selection down one item

Parameters

None

StackBox.DropN

Deletes the selected item, and all items higher (newer) on the stack

Parameters

None

StackBox.DupN

Duplicates the selected item, and all items higher (newer) on the stack

Parameters

None

StackBox.EE

Adds an exponential symbol to the console, if one does not already exist, and the number format allows it.

Parameters

None

StackBox.Enter

If a value is on the console, pushes that value to the stack. If an item on the stack is

selected, duplicates that item and pushes it to the top of the stack. Otherwise, duplicates the top item on the stack.

Parameters

None

StackBox.Explode

Explodes the top item on the stack. Often thought of as being 'Undo'.

Parameters

x No restrictions

StackBox.LeftArrow

Moves the stack selection left one item

Parameters

None

StackBox.ModeApprox

Sets the stack number format to approximate

Parameters

None

StackBox.ModeAuto

Sets the stack number format to auto (i.e. automatically switch between normal and scientific notation)

Parameters

None

StackBox.ModeBin

Sets the stack number format to binary based-integers. Most recently used number of bits will be maintained.

Parameters

None

StackBox.ModeBool

Sets the stack number format to 1-bit boolean based-integers. Most recently used base will be maintained.

Parameters

None

StackBox.ModeByte

Sets the stack number format to 8-bit based-integers. Most recently used base will be maintained.

Parameters

None

StackBox.ModeDec

Sets the stack number format to decimal based-integers. Most recently used number of bits will be maintained.

Parameters

None

StackBox.ModeDword

Sets the stack number format to 32-bit based-integers. Most recently used base will be maintained.

Parameters

None

StackBox.ModeEng

Sets the stack number format to engineering

Parameters

None

StackBox.ModeExact

Sets the stack number format to exact (i.e. use the perfect number engine)

Parameters

None

StackBox.ModeFixed

Sets the stack number format to fixed number of decimal places (as defined by ModeSigFig#)

Parameters

None

StackBox.ModeFloat

Sets the stack number format to a floating number of decimal places (as defined by ModeSigFig#)

Parameters

None

StackBox.ModeHex

Sets the stack number format to hexadecimal based-integers. Most recently used number of bits will be maintained.

Parameters

None

StackBox.ModeQword

Sets the stack number format to 64-bit based-integers. Most recently used base will be maintained.

Parameters

None

StackBox.ModeSci

Sets the stack number format to scientific (as opposed to engineering or auto)

Parameters

None

StackBox.ModeScientific

Sets the calculator mode to scientific (as opposed to based-integer)

Parameters

None

StackBox.ModeSigFig0

Sets the stack number format to 0 significant digits

Parameters

None

StackBox.ModeSigFig1

Sets the stack number format to 1 significant digit

Parameters

None

StackBox.ModeSigFig2

Sets the stack number format to 2 significant digits

Parameters

None

StackBox.ModeSigFig3

Sets the stack number format to 3 significant digits

Parameters

None

StackBox.ModeSigFig4

Sets the stack number format to 4 significant digits

Parameters

None

StackBox.ModeSigFig5

Sets the stack number format to 5 significant digits

Parameters

None

StackBox.ModeSigFig6

Sets the stack number format to 6 significant digits

Parameters

None

StackBox.ModeSigFig7

Sets the stack number format to 7 significant digits

Parameters

None

StackBox.ModeSigFig8

Sets the stack number format to 8 significant digits

Parameters

None

StackBox.ModeSigFig9

Sets the stack number format to 9 significant digits

Parameters

None

StackBox.ModeSigFig10

Sets the stack number format to 10 significant digits

Parameters

None

StackBox.ModeSigFig11

Sets the stack number format to 11 significant digits

Parameters

None

StackBox.ModeSigFig12

Sets the stack number format to 12 significant digits

Parameters

None

StackBox.ModeSigFig13

Sets the stack number format to 13 significant digits

Parameters

None

StackBox.ModeSigFig14

Sets the stack number format to 14 significant digits

Parameters

None

StackBox.ModeSigFig15

Sets the stack number format to 15 significant digits

Parameters

None

StackBox.ModeToggleScientific

Toggles between scientific and based-integer modes

Parameters

None

StackBox.ModeWord

Sets the stack number format to 16-bit based-integers. Most recently used base will be maintained.

Parameters

None

StackBox.OpenFile

Displays the file open dialog

Parameters

None

StackBox.RightArrow

Moves the stack selection right one item

Parameters

None

StackBox.Roll3Down

Rolls the top three items on the stack downward

Parameters

z No restrictions

y No restrictions

x No restrictions

StackBox.Roll3Up

Rolls the top three items on the stack upward

Parameters

z No restrictions

y No restrictions

x No restrictions

StackBox.Roll4Down

Rolls the top four items on the stack downward

Parameters

t No restrictions

z No restrictions

y No restrictions

x No restrictions

StackBox.Roll4Up

Rolls the top four items on the stack upward

Parameters

t No restrictions

z No restrictions

y No restrictions

x No restrictions

StackBox.RollAllDown

Rolls all the items on the stack downward

Parameters

None

StackBox.RollAllUp

Rolls all the items on the stack upward

Parameters

None

StackBox.RollNDown

Rolls the selected item, and all items higher (newer) on the stack downward

Parameters

None

StackBox.RollNUp

Rolls the selected item, and all items higher (newer) on the stack upward

Parameters

None

StackBox.RollToTop

Rolls the entire stack downward until the selected item is highest (newest). Has no effect if nothing is selected.

Parameters

None

StackBox.RollToTopUndo

Rolls the entire stack upward the same number of steps as the previous RollToTop rolled down. The item that was on top of the stack prior to RollToTopUndo will be selected and scrolled into view. Has no effect if RollToTop has not been called, or the most recent call to RollToTop had no effect.

Parameters

None

StackBox.Share

Displays the file share dialog

Parameters

None

StackBox.Swap2

Swaps the top two items on the stack

Parameters

y No restrictions

x No restrictions

StackBox.UpArrow

Moves the stack selection up one item

Parameters

None

Math10x.Action

Computes 10^x

Parameters

x No restrictions

Math2x.Action

Computes 2^x

Parameters

x No restrictions

MathAbs.Action

Computes $|x|$

Parameters

x No restrictions

MathACosD.Action

Computes $cos_{deg}^{-1}(x)$

Parameters

x No restrictions

MathACosh.Action

Computes $cosh^{-1}(x)$

Parameters

x No restrictions

MathACosR.Action

Computes $cos^{-1}(x)$

Parameters

x No restrictions

MathACotD.Action

Computes $cot_{deg}^{-1}(x)$

Parameters

x No restrictions

MathACotR.Action

Computes $cot^{-1}(x)$

Parameters

x No restrictions

MathACscD.Action

Computes $csc_{deg}^{-1}(x)$

Parameters

x No restrictions

MathACscR.Action

Computes $csc^{-1}(x)$

Parameters

x No restrictions

MathAdd.Action

Computes $y + x$

Parameters

y No restrictions

x If x is a percentage, will create a MathAddPercentage. Otherwise, no restrictions

MathAddPercentage.Action

Computes $y + x\%$

Parameters

y No restrictions

x Must be a percentage

MathAnd.Action

Computes bitwise $y\ and\ x$

Parameters

y Must be a based-integer

x Must be a based-integer

MathASecD.Action

Computes $sec_{deg}^{-1}(x)$

Parameters

x No restrictions

MathASecR.Action

Computes $sec^{-1}(x)$

Parameters

x No restrictions

MathASinD.Action

Computes $sin_{deg}^{-1}(x)$

Parameters

x No restrictions

MathASinh.Action

Computes $sinh^{-1}(x)$

Parameters

x No restrictions

MathASinR.Action

Computes $sin^{-1}(x)$

Parameters

x No restrictions

MathATanD.Action

Computes $tan_{deg}^{-1}(x)$

Parameters

x No restrictions

MathATanh.Action

Computes $tanh^{-1}(x)$

Parameters

x No restrictions

MathATanR.Action

Computes $tan^{-1}(x)$

Parameters

x No restrictions

MathBool.Action

Converts x to a 1-bit boolean based-integer. Returns false if x is zero, true if x is non-zero.

Parameters

x No restrictions

MathCeiling.Action

Computes $\lceil x \rceil$

Parameters

x No restrictions

MathConj.Action

Computes complex conjugate $conj(x)$

Parameters

x No restrictions

MathCosD.Action

Computes $cos(x°)$

Parameters

x No restrictions

MathCosh.Action

Computes $cosh(x)$

Parameters

x No restrictions

MathCosR.Action

Computes $cos(x)$

Parameters

x No restrictions

MathCotD.Action

Computes $cot(x°)$

Parameters

x No restrictions

MathCotR.Action

Computes $cot(x)$

Parameters

x No restrictions

MathCritical.Action

Computes $critical(f, v, x)$. Critical points of a function are where the derivative of the function is zero. f must be continuous and real near x.

Parameters

f A function of variable v. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

v Variable

x Seed value to start the computation. Must not be a complex number. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

— or —

f A MathCritical, MathDerivative, MathWhere, or MathZero. f, v, and x will be extracted from the previous computation.

— or —

f A MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero. f and v will be extracted from the previous computation.

x Seed value to start the computation. Must not be a complex number. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

MathCscD.Action

Computes $csc(x°)$

Parameters

x No restrictions

MathCscR.Action

Computes $csc(x)$

Parameters

x No restrictions

MathCube.Action

Computes x^3

Parameters

x No restrictions

MathCubeRoot.Action

Computes $\sqrt[3]{x}$

Parameters

x No restrictions

MathDeltaPercent.Action

Computes the percent change from y to x.

Parameters

y Must not be a complex number

x Must not be a complex number

MathDerivative.Action

Computes $\frac{d}{dv}(f, v, x)$.

Parameters

f A function of variable v. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

v Variable

x Seed value to start the computation. Must not be a complex number. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

— or —

f A MathCritical, MathDerivative, MathWhere, or MathZero. f, v, and x will be extracted from the previous computation.

— or —

f A MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero. f and v will be extracted from the previous computation.

x Seed value to start the computation. Must not be a complex number. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

MathDivide.Action

Computes $\frac{y}{x}$

Parameters

y No restrictions

x No restrictions

MathDivInvert.Action

Computes $\frac{1}{x}$

Parameters

x No restrictions

MathDivisors.Action

Computes how many integers divide x.

Parameters

x Must be a perfect integer (not a based-integer)

MathDMS.Action

Adds a degrees, minutes, or seconds symbol to the console.

Parameters

None

MathDoubleFactorial.Action

Computes x!!

Parameters

x Must be a perfect integer (not a based-integer)

MathE.Action

Pushes mathematical constant e to the stack. If a value is on the console, it will be multiplied by e.

Parameters

None

MathEquals.Action

Computes $y = x$

Parameters

y No restrictions

x No restrictions

MathEulerPhi.Action

Computes $\phi(x)$

Parameters

x Must be a perfect integer (not a based-integer)

MathExp.Action

Computes e^x

Parameters

x No restrictions

MathFactor.Action

Computes the prime factorization of x

Parameters

x Must be a perfect integer (not a based-integer)

MathFactorial.Action

Computes x!

Parameters

x No restrictions

MathFloor.Action

Computes $\lfloor x \rfloor$

Parameters

x No restrictions

MathFPart.Action

Computes the fractional part of x

Parameters

x No restrictions

MathGamma.Action

Computes $\Gamma(x)$

Parameters

x No restrictions

MathGCD.Action

Computes $\gcd(y, x)$

Parameters

y Must be a perfect integer (not a based-integer)

x Must be a perfect integer (not a based-integer)

MathGreaterThan.Action

Computes $y > x$

Parameters

y No restrictions

x No restrictions

MathGreaterThanEqual.Action

Computes $y \geq x$

Parameters

y No restrictions

x No restrictions

MathI.Action

Pushes mathematical constant i to the stack. If a value is on the console, it will be multiplied by i.

Parameters

None

MathI.Action2

Computes $y + x \cdot i$

Parameters

y No restrictions

x No restrictions

MathIf.Action

Computes $\text{if}(z, y, x)$, i.e if z is non-zero return y, else return x.

Parameters

z No restrictions

y No restrictions

x No restrictions

MathImag.Action

Computes the imaginary part of x

Parameters

x No restrictions

MathInt8.Action

Converts x to an 8-bit based-integer. Rounds and truncates high bytes if necessary.

Parameters

x Must not be a complex number.

MathInt16.Action

Converts x to a 16-bit based-integer. Rounds and truncates high bytes if necessary.

Parameters

x Must not be a complex number.

MathInt32.Action

Converts x to a 32-bit based-integer. Rounds and truncates high bytes if necessary.

Parameters

x Must not be a complex number.

MathInt64.Action

Converts x to a 64-bit based-integer. Rounds if necessary.

Parameters

x Must not be a complex number.

MathIntegral.Action

Computes $\int(f, v, y, x)$

Parameters

f A function of variable v. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

v Variable

y Lower bound to start the computation. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

x Upper bound to start the computation. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

— or —

f A MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero. f and v will be extracted from the previous computation.

y Lower bound to start the computation. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

x Upper bound to start the computation. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

— or —

f MathIntegral to use as the inner integral. Will create a MathDoubleIntegral by extracting the function, inner variable and inner bounds from the original MathIntegral.

v Outer variable

y Outer lower bound to start the computation. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

x Outer upper bound to start the computation. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

MathIPart.Action

Computes the integer part of x

Parameters

x No restrictions

MathLCM.Action

Computes $\mathrm{lcm}(y, x)$

Parameters

y Must be a perfect integer (not a based-integer)

x Must be a perfect integer (not a based-integer)

MathLessThan.Action

Computes $y < x$

Parameters

y No restrictions

x No restrictions

MathLessThanEqual.Action

Computes $y \leq x$

Parameters

y No restrictions

x No restrictions

MathLn.Action

Computes $\ln(x)$

Parameters

x No restrictions

MathLog.Action

Computes $\log(x)$

Parameters

x No restrictions

MathLog2.Action

Computes $\log_2(x)$

Parameters

x No restrictions

MathLogn.Action

Computes $\log_x(y)$

Parameters

y No restrictions

x If x is not a perfect integer, MathLogn will be converted into $\frac{\ln(y)}{\ln(x)}$

MathLShift.Action

Computes bitwise $y \ll x$

Parameters

y Must be a based-integer

x Must be a based-integer

MathMag.Action

Computes the magnitude of x

Parameters

x No restrictions

MathMagnitudePhase.Action

Computes $y\angle x$ (in radians)

Parameters

y Must not be a complex number

x Must not be a complex number

MathMagnitudePhaseD.Action

Computes $y\angle x°$

Parameters

y Must not be a complex number

x Must not be a complex number

MathMax.Action

Computes $\max(y, x)$

Parameters

y Must not be a complex number

x Must not be a complex number

MathMemoryStore.LastX

Recalls the last value to be in the x register

Parameters

None

MathMemoryStore.MemoryClear

Clears the memory register

Parameters

None

MathMemoryStore.MemoryRecall

Recalls the memory register

Parameters

None

MathMemoryStore.MemoryStore

Stores the selected value, or the top of the stack, to the memory register

Parameters

None

MathMin.Action

Computes $\min(y, x)$

Parameters

y Must not be a complex number

x Must not be a complex number

MathMod.Action

Computes $\mathrm{mod}(y, x)$

Parameters

y Must be a perfect integer (not a based-integer)

x Must be a perfect integer (not a based-integer)

— or —

y Must be a based-integer

x Must be a based-integer

MathMoebiusMu.Action

Computes $\mu(x)$

Parameters

x Must be a perfect integer (not a based-integer)

MathMultichoose.Action

Computes $\mathrm{C}_{\mathrm{multi}}(y, x)$

Parameters

y Must be a perfect integer (not a based-integer)

x Must be a perfect integer (not a based-integer)

MathMultiply.Action

Computes $y \cdot x$

Parameters

y No restrictions

x No restrictions

MathNand.Action

Computes bitwise $y\ nand\ x$

Parameters

y Must be a based-integer

x Must be a based-integer

MathNCR.Action

Computes $\mathrm{C}(y, x)$

Parameters

y Must be a perfect integer (not a based-integer)

x Must be a perfect integer (not a based-integer)

MathNegation.Action

Computes -x

Parameters

x No restrictions

MathNor.Action

Computes bitwise $y\ nor\ x$

Parameters

y Must be a based-integer

x Must be a based-integer

MathNot.Action

Computes bitwise $not\ x$

Parameters

x Must be a based-integer

MathNotEqual.Action

Computes $y \neq x$

Parameters

y No restrictions

x No restrictions

MathNPR.Action

Computes $\mathrm{P}(y, x)$

Parameters

y Must be a perfect integer (not a based-integer)

x Must be a perfect integer (not a based-integer)

MathNRoot.Action

Computes $\sqrt[x]{y}$

Parameters

y No restrictions

x If x is not a perfect integer, MathNRoot will be converted into $y^{\frac{1}{x}}$

MathOr.Action

Computes bitwise $y\ or\ x$

Parameters

y Must be a based-integer

x Must be a based-integer

MathPercent.Action

Computes $x\%$

Parameters

x Must not be a complex number

MathPercentTotal.Action

Computes the percentage of y that is x.

Parameters

y Must not be a complex number

x Must not be a complex number

MathPhase.Action

Computes the phase part of x (in radians)

Parameters

x No restrictions

MathPhaseD.Action

Computes the phase part of x (in degrees)

Parameters

x No restrictions

MathPI.Action

Pushes mathematical constant π to the stack. If a value is on the console, it will be multiplied by π.

Parameters

None

MathPower.Action

Computes y^x

Parameters

y No restrictions

x No restrictions

MathPower.Action2

Computes x^y

Parameters

y No restrictions

x No restrictions

MathPowInvert.Action

Computes x^{-1}

Parameters

x No restrictions

MathPrime.Action

Computes the x-th largest prime

Parameters

x Must be a perfect integer (not a based-integer) between 1 and 6542

MathPrimePi.Action

Computes the number of primes less than or equal to x

Parameters

x Must not be a complex number. Must be less than or equal to 65521.

MathProduct.Action

Computes $\prod(f, v, y, x)$

Parameters

f — A function of variable v. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

v — Variable

y — Lower bound to start the computation. Must be a perfect integer (not a based-integer). Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

x — Upper bound to start the computation. Must be a perfect integer (not a based-integer). Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

— or —

f — A MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero. f and v will be extracted from the previous computation.

y — Lower bound to start the computation. Must be a perfect integer (not a based-integer). Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

x — Upper bound to start the computation. Must be a perfect integer (not a based-integer). Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

MathRandom.Action

Puts a random number on the console. If there was already a number on the console, it will be replaced with a random number between it and zero. Otherwise, creates a random floating-point number between zero and one.

Parameters

None

MathRationalize.Action

Converts y to a fraction, with maximum denominator x

Parameters

y — Must not be a complex number

x — Must not be a complex number and must be greater than or equal to 1

MathReal.Action

Computes the real part of x

Parameters

x — No restrictions

MathRound.Action

Computes $[x]$

Parameters

x — No restrictions

MathRShift.Action

Computes bitwise $y \gg x$

Parameters

y Must be a based-integer

x Must be a based-integer

MathSecD.Action

Computes $sec(x°)$

Parameters

x No restrictions

MathSecR.Action

Computes $sec(x)$

Parameters

x No restrictions

MathSign.Action

Computes $sign(x)$

Parameters

x No restrictions

MathSinD.Action

Computes $sin(x°)$

Parameters

x No restrictions

MathSinh.Action

Computes $sinh(x)$

Parameters

x No restrictions

MathSinR.Action

Computes $sin(x)$

Parameters

x No restrictions

MathSqrt.Action

Computes $\sqrt{x}$

Parameters

x No restrictions

MathSquare.Action

Computes x^2

Parameters

x No restrictions

MathStatistics.AppendXY

Adds a new data point with x as the independent data and y as the dependent data

Parameters

y Must not be a complex number

x Must not be a complex number

MathStatistics.AppendYX

Adds a new data point with y as the independent data and x as the dependent data

Parameters

y Must not be a complex number

x Must not be a complex number

MathStatistics.CalcExponentialRegression

Populates the a, b, r, and r^2 registers based on the best-fit exponential regression.

Parameters

None

MathStatistics.CalcLinearRegression

Populates the a, b, r, and r^2 registers based on the best-fit linear regression.

Parameters

None

MathStatistics.CalcLogarithmicRegression

Populates the a, b, r, and r^2 registers based on the best-fit logarithmic regression.

Parameters

None

MathStatistics.CalcPowerRegression

Populates the a, b, r, and r^2 registers based on the best-fit power regression.

Parameters

None

MathStatistics.CalculateStatistics

Populates the statistics registers based on selected item, and all items higher (newer) on the stack, using the real part of the answer as the independent data and the imaginary part as the dependent data.

Parameters

None

MathStatistics.ClearStatistics

Clears all the statistics registers

Parameters

None

MathStatistics.RemoveXY

Removes a data point with x as the independent data and y as the dependent data. An exact match for this point must have been previously entered.

Parameters

y Must not be a complex number

x Must not be a complex number

MathStatistics.RemoveYX

Removes a data point with y as the independent data and x as the dependent data. An exact match for this point must have been previously entered.

Parameters

y Must not be a complex number

x Must not be a complex number

MathStatistics.StatCount

Pushes the n statistics variable to the stack

Parameters

None

MathStatistics.StatLinRegA

Pushes the a statistics variable to the stack

Parameters

None

MathStatistics.StatLinRegB

Pushes the b statistics variable to the stack

Parameters

None

MathStatistics.StatLinRegR

Pushes the r statistics variable to the stack

Parameters

None

MathStatistics.StatLinRegRSquared

Pushes the r^2 statistics variable to the stack

Parameters

None

MathStatistics.StatMaxX

Pushes the x_{max} statistics variable to the stack

Parameters

None

MathStatistics.StatMaxY

Pushes the y_{max} statistics variable to the stack

Parameters

None

MathStatistics.StatMeanX

Pushes the $\overline{x}$ statistics variable to the stack

Parameters

None

MathStatistics.StatMeanY

Pushes the $\overline{y}$ statistics variable to the stack

Parameters

None

MathStatistics.StatMedianX

Pushes the $\hat{x}$ statistics variable to the stack

Parameters

None

MathStatistics.StatMedianY

Pushes the $\hat{y}$ statistics variable to the stack

Parameters

None

MathStatistics.StatMinX

Pushes the x_{min} statistics variable to the stack

Parameters

None

MathStatistics.StatMinY

Pushes the y_{min} statistics variable to the stack

Parameters

None

MathStatistics.StatRegression

Creates a macro based on the current regression model that takes x values as input and projects y values as output.

Parameters

None

MathStatistics.StatRegressionInverse

Creates a macro based on the current regression model that takes y values as input and projects x values as output.

Parameters

None

MathStatistics.StatRegressionWhere

Creates a MathWhere equation based on the current regression model that takes x values as input and projects y values as output.

Parameters

x Must not be a complex number

MathStatistics.StatRegressionWhereInverse

Creates a MathWhere equation based on the current regression model that takes y values as input and projects x values as output.

Parameters

x Must not be a complex number

MathStatistics.StatSampleStandardDevX

Pushes the sx statistics variable to the stack

Parameters

None

MathStatistics.StatSampleStandardDevY

Pushes the sy statistics variable to the stack

Parameters

None

MathStatistics.StatStandardDevX

Pushes the σx statistics variable to the stack

Parameters

None

MathStatistics.StatStandardDevY

Pushes the σy statistics variable to the stack

Parameters

None

MathStatistics.StatSumX

Pushes the $\sum x$ statistics variable to the stack

Parameters

None

MathStatistics.StatSumX2

Pushes the $\sum x^2$ statistics variable to the stack

Parameters

None

MathStatistics.StatSumXY

Pushes the $\sum xy$ statistics variable to the stack

Parameters

None

MathStatistics.StatSumY

Pushes the $\sum y$ statistics variable to the stack

Parameters

None

MathStatistics.StatSumY2

Pushes the $\sum y^2$ statistics variable to the stack

Parameters

None

MathStatistics.StatWeightedMean

Pushes the $\overline{x}_w$ statistics variable to the stack

Parameters

None

MathStatistics.StatWeightedSampleStandardDev

Pushes the sx_w statistics variable to the stack

Parameters

None

MathStatistics.StatWeightedStandardDev

Pushes the σx_w statistics variable to the stack

Parameters

None

MathSubFactorial.Action

Computes $!x$

Parameters

x Must be a perfect integer (not a based-integer)

MathSubtract.Action

Computes $y - x$

Parameters

y No restrictions

x If x is a percentage, will create a MathSubtractPercentage. Otherwise, no restrictions

MathSubtractPercentage.Action

Computes $y - x\%$

Parameters

y No restrictions

x Must be a percentage

MathSumDivisors.Action

Computes the sum of integers that divide x.

Parameters

x Must be a perfect integer (not a based-integer)

MathSummation.Action

Computes $\sum(f, v, y,\ x)$

Parameters

f A function of variable v. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

v Variable

y Lower bound to start the computation. Must be a perfect integer (not a based-integer). Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

x Upper bound to start the computation. Must be a perfect integer (not a based-integer). Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct,

MathSummation, MathWhere, or MathZero.

— or —

f A MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero. f and v will be extracted from the previous computation.

y Lower bound to start the computation. Must be a perfect integer (not a based-integer). Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

x Upper bound to start the computation. Must be a perfect integer (not a based-integer). Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

MathTanD.Action

Computes $tan(x°)$

Parameters

x No restrictions

MathTanh.Action

Computes $tanh(x)$

Parameters

x No restrictions

MathTanR.Action

Computes $tan(x)$

Parameters

x No restrictions

MathToDMS.Action

Converts to degrees-minutes-seconds format

Parameters

x Must not be a complex number. Must be between -2147483648 and 2147483647.

MathToMagnitudePhase.Action

Converts complex numbers to polar $r\angle\theta$ format (in radians).

Parameters

x No restrictions

MathToMagnitudePhaseD.Action

Converts complex numbers to polar $r\angle\theta°$ format (in degrees).

Parameters

x No restrictions

MathVariableStore.Action

Store value y into variable x

Parameters

y No restrictions

x Must be a variable

MathWhere.Action

Computes $f(v)|v = x$.

Parameters

f A function of variable v. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct,

MathSummation, MathWhere, or MathZero.

v Variable

x Seed value to start the computation. Must not be a complex number. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

— or —

f A MathCritical, MathDerivative, MathWhere, or MathZero. f, v, and x will be extracted from the previous computation.

— or —

f A MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero. f and v will be extracted from the previous computation.

x Seed value to start the computation. Must not be a complex number. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

— or —

f MathWhere to which to add an additional variable constraint.

v New variable to add as an additional constraint. If v was already a constraint on f, that constraint will be replaced.

x Seed value for the new constraint. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

MathXnor.Action

Computes bitwise $y\ xnor\ x$

Parameters

y Must be a based-integer

x Must be a based-integer

MathXor.Action

Computes bitwise $y\ xor\ x$

Parameters

y Must be a based-integer

x Must be a based-integer

MathZero.Action

Computes $zero(f, v, x)$. f must be continuous and real near x.

Parameters

f A function of variable v. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

v Variable

x Seed value to start the computation. Must not be a complex number. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

— or —

f A MathCritical, MathDerivative, MathWhere, or MathZero. f, v, and x will be extracted from the previous computation.

— or —

f A MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero. f and v will be extracted from the previous computation.

x Seed value to start the computation. Must not be a complex number. Cannot contain other MathCritical, MathDerivative, MathIntegral, MathProduct, MathSummation, MathWhere, or MathZero.

MacroBox.ApplyMoveForward

Update the current cell to the value on the console, and advance the selection to the next cell

Parameters

None

MacroBox.Backspace

Deletes one character from the console

Parameters

None

MacroBox.Cancel

Exit macro/edit mode without making any changes to the stack

Parameters

None

MacroBox.Clear

Clears the text on the console

Parameters

None

MacroBox.Decimal

Adds a decimal to the console, if one does not already exist, and the number format allows it.

MacroBox.Digit0

Adds a zero to the console (if the number format allows it)

Parameters

None

MacroBox.Digit1

Adds a one to the console (if the number format allows it)

Parameters

None

MacroBox.Digit2

Adds a two to the console (if the number format allows it)

Parameters

None

MacroBox.Digit3

Adds a three to the console (if the number format allows it)

Parameters

None

MacroBox.Digit4

Adds a four to the console (if the number format allows it)

Parameters

None

MacroBox.Digit5

Adds a five to the console (if the number format allows it)

Parameters

None

MacroBox.Digit6

Adds a six to the console (if the number format allows it)

Parameters

None

MacroBox.Digit7

Adds a seven to the console (if the number format allows it)

Parameters

None

MacroBox.Digit8

Adds an eight to the console (if the number format allows it)

Parameters

None

MacroBox.Digit9

Adds a nine to the console (if the number format allows it)

Parameters

None

MacroBox.DigitA

Adds a hexadecimal ‘A’ to the console (if the number format allows it)

Parameters

None

MacroBox.DigitB

Adds a hexadecimal ‘B’ to the console (if the number format allows it)

Parameters

None

MacroBox.DigitC

Adds a hexadecimal ‘C’ to the console (if the number format allows it)

Parameters

None

MacroBox.DigitD

Adds a hexadecimal ‘D’ to the console (if the number format allows it)

Parameters

None

MacroBox.DigitE

Adds a hexadecimal ‘E’ to the console (if the number format allows it)

Parameters

None

MacroBox.DigitF

Adds a hexadecimal ‘F’ to the console (if the number format allows it)

Parameters

None

MacroBox.DiscardMoveBackward

Discard the value currently on the console, and move the selection back to the previous cell

Parameters

None

MacroBox.EditOK

Update the original stack entry to the current result and exit edit mode

Parameters

None

MacroBox.EE

Adds an exponential symbol to the console, if one does not already exist, and the number format allows it.

Parameters

None

MacroBox.Freeze

Freeze the selected cell so that you will no longer be prompted to fill in its value

Parameters

None

MacroBox.MacroOK

Push the current result to the stack and close macro mode

Parameters

None

MacroBox.MacroPush

Push the current result to the stack, but remain in macro mode

Parameters

None

MacroBox.MemoryRecall

Recall the memory register to the console.

Parameters

None

MacroBox.Negative

Negates the value on the console

Parameters

None

MacroBox.ShowEdit

Switch from normal stack entry into edit mode

Parameters

None

MacroBox.ShowMacro

Switch from normal stack entry into macro mode

Parameters

None

MacroBox.Split

Split the selected cell, so that it is no longer grouped with matching cells

Parameters

None

MacroBox.Zoom

Cycle through available zoom levels to display the macro

Parameters

None

Appendix 2: Document History

Date	Version	Description
8/27/2017	3.0.0	Initial version
1/10/2018	3.0.1	Support for double integrals. Fixed other minor typographical errors.
6/1/2018	3.1.0	Support for functions and subroutines in values section. Added boolean and comparison operators.
8/31/2018	3.1.1	Change the alternate view suffixes via attributes on <stack>